

FIG. 1

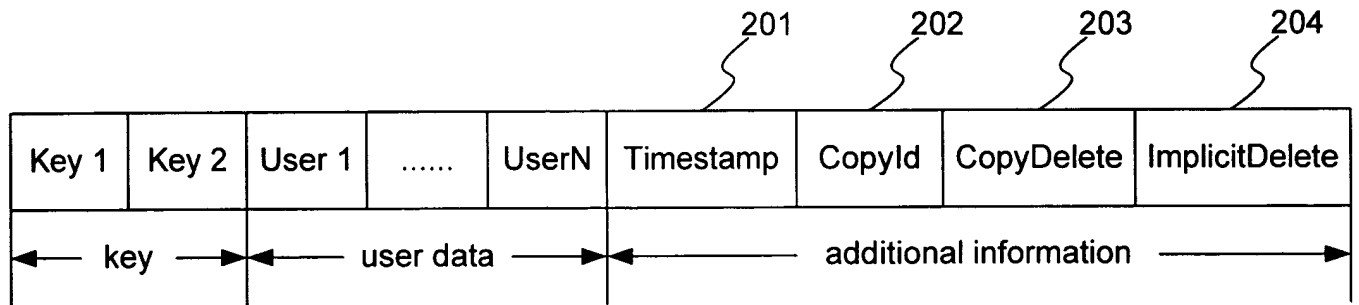


FIG. 2

Copy A	Copy B
//user insert *ins X[1a] ---> m1 (301)	//user insert *ins X[2b] ---> m2 (302)
	//user delete *del X[2b] ---> m3 (303)
//insert conflict m2 ---> implDel X[1a] ---> m4 (304) ins X[2b] (305)	
//propagated delete m3 ---> del X[2b] (306)	//propagated insert m1 ---> ins X[1a] (307)
	//propagated implicit delete m4 ---> del X[1a] (308)

FIG. 3

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (401)	//user insert *ins X[2b] ---> m2 (402)	
		//insert from 'b' m2 ---> ins X[2b] (403)
		//user delete *del X[2b] ---> m3 (404)
//delete conflict: $2b > 1a$ m3 ---> impleDel X[1a] ---> m4 (405) deltomb add X[2b] (406)	//non-conflict delete from 'c' m3 ---> del X[2b] (407)	
	//late arriving insert from 'a' m1 ---> ins X[1a] (408)	//late arriving insert from 'a' m1 ---> ins X[1a] (409)
//late arriving insert from 'b' m2 ---> X[2b] in deltomb (410)		
	//implicit delete from 'a' m4 ---> del X[1a] (411)	//implicit delete from 'a' m4 ---> del X[1a] (412)

FIG. 4

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (501)	//user insert *ins X[2b] ---> m2 (502)	
		//insert from 'a' m1 ---> ins X[1a] (503)
		//user delete *del X[1b] ---> m3 (504)
//non-conflict delete from 'c' m3 ---> del X[1a] (505)	//delete conflict: 1a<2b m3 ---> deltomb add X[1a] (506)	
//late arriving insert from 'b' m2 ---> ins X[2b] (507)		//late arriving insert from 'b' m2 ---> ins X[2b] (508)
		//user delete *del X[2b] ---> m4 (509)
//non-conflict delete from 'c' m4 ---> del X[2b] (510)	//non-conflict delete from 'c' m4 ---> del X[2b] (511)	
	//late arriving insert from 'a' m1 ---> X[1a] in deltomb (512)	

FIG. 5

5/17

Copy A	Copy B	Copy C
<i>//user insert</i> *ins X[1a] ---> m1 (601)		
	<i>//propagated insert</i> m1 ---> ins X[1a] (602)	
	<i>//user delete</i> *del X[1a] ---> m2 (603)	
<i>//non-conflict delete from</i> <i>'b'</i> m2 ---> del X[1a] (604)		<i>//delete conflict: not found</i> m2 ---> deltomb add X[1a] (605)
		<i>//late arriving insert from 'a'</i> m1 ---> X[1a] in deltomb (606)

FIG. 6

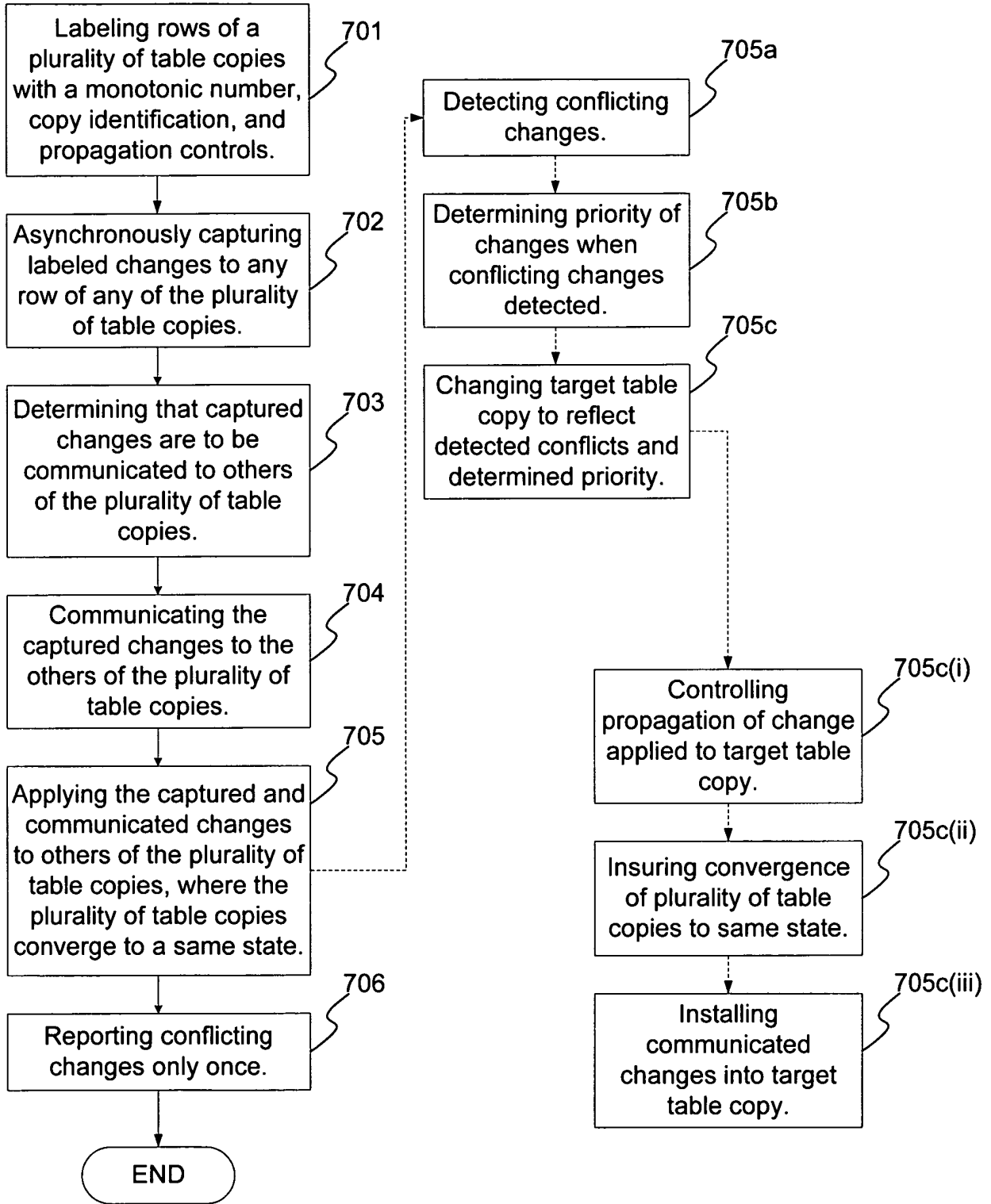


FIG. 7

```

If (log.tableId in CopyTables)
  switch (log.type)
    case Insert:
      if(log.new.CopyId==this Copy)
        sendInsertMsg (log.tableId,
          log.new.keyCols, log.new.nonKeyCols,
          log.new.Timestamp, log.new.CopyId);
    case Delete:
      if (log.old.CopyDelete=='N')
        sendDeleteMsg (log.tableId,
          log.old.keyCols,
          log.old.Timestamp, log.old.CopyId);
    case Update:
    case KeyUpdate:
      if (log.new.ImplicitDelete=='Y')
        sendDeleteMsg (log.tableId,
          log.old.keyCols,
          log.old.Timestamp, log.old.CopyId);
      if ((log.new.CopyDelete=='N') &&
        (log.new.CopyId==thisCopy))
      if (log.type==Update)
        sendUpdateMsg (log.tableId,
          log.new.keyCols, log.new.nonKeyCols,
          log.new.Timestamp, log.new.CopyId,
          log.old.Timestamp, log.old.CopyId);
      else
        sendKeyUpdMsg (log.tableId,
          log.new.KeyCols, Log.new.nonKeyCols,
          log.new.Timestamp, log.new.CopyId,
          log.old.KeyCols, log.old.Timestamp,
          log.old.CopyId);

```

FIG. 8

8/17

```
while (UCTtime() < delmsg.old.Timestamp)
    sleep (delmsg.Timestamp - UCTtime()); //wait for future
deleteConflict = reportConflict = FALSE; //default values
DECLARE delcur CURSOR FOR
    SELECT Timestamp, CopyId
    FROM delmsg.tableId target    -- the target table
    WHERE delmsg.keyCols = target.keyCols
    FOR UPDATE OF Timestamp, CopyId, CopyDelete, ImplicitDelete;
OPEN CURSOR delcur;
FETCH delcur INTO Timestamp, CopyId;
if (sqlcode == SQL_OK )
    ImplicitDeleteFlag='N';
    if ((Timestamp != delmsg.Timestamp) ||
        (CopyId != delmsg.CopyId)) //unmatched Timestamp
        deleteConflict = TRUE;
    if ((Timestamp <= delmsg.Timestamp) ||
        (Timestamp==delmsg.Timestamp)&& (CopyId<=delmsg.CopyId))
        //row time <= message time=>must delete row
    if ((CopyId == thisCopy) &&
        (Timestamp!=delmsg.Timestamp || CopyId!=delmsg.CopyId))
        ImplicitDeleteFlag = 'Y';
    UPDATE delmsg.TableId
        SET ImplicitDelete = ImplicitDeleteFlag,
            CopyDelete = 'Y'          -- prevent capture
            Timestamp = '0001-01-01.01' -- prevent trigger
        WHERE CURRENT OF delcur;
    DELETE FROM delmsg.tableId
        WHERE CURRENT OF delcur;
    //row time>message time => don't delete row
    if (CopyId == this Copy)
        reportConflict = TRUE;
else // no row matched message row
    deleteConflict = TRUE;
    if (CopyId == thisCopy)
        reportConflict = TRUE;
    if (deleteConflict)
        if (delmsg.CopyId != thisCopy)
            INSERT INTO deltomb VALUES
                (delmsg.tableId, delmsg.keyCols,
                 delmsg.Timestamp, delmsg.CpyId);
    if (reportConflict )
        logConflict ("DELETE", delmsg);
```

FIG. 9

9/17

```
while (UCTtime() < insmsg.new.Timestamp)
  sleep (insmsg.Timestamp - UCTtime()); //wait for future
  checkDeltomb = FALSE; implDelete = 'N';
  // attempt a simple insert of the message row
  INSERT INTO insmsg.tableId VALUES
    (insmsg.keyCols, insmsg.nonKeyCols,
     insmsg.Timestamp, insmsg.CopyId);
  if (sqlcode == 'key violation')
    // fetch existing row for possible implicit delete
    DECLARE inscur CURSOR FOR
      SELECT Timestamp, CopyId
      FROM insmsg.tableId target
      WHERE insmsg.keyCols = target.keyCols
      FOR UPDATE OF nonKeyCols, "Timestamp", "CopyId";
    OPEN CURSOR inscur;
    FETCH inscur INTO Timestamp, CopyId;
    if (Timestamp < insmsg.Timestamp ||
        (Timestamp == insmsg.Timestamp &&
         CopyId < insmst.CopyId))
      //Message row is winner.. Implicit delete of existing row
      checkDeltomb = TRUE;
      if (CopyId == thisCopy)
        implDelete = 'Y';
      UPDATE insmsg.tableId
        SET nonKeyCols = insmsg.nonKeyCols,
            Timestamp = insmsg.Timestamp,
            CopyId = insmsg.CopyId,
            ImplicitDelete = implDelete
        WHERE CURRENT OF inscur;
      else if (CopyId == thisCopy)
        // 'key violation' and existing row is the winner
        logConflict ("INSERT", insmsg);
    else // no 'key violation'
      checkDeltomb = TRUE;
```

FIG. 10

10/17

```
if (checkDeltomb)
    SELECT 1 FROM del tomb
        WHERE tableId = insmsg.tableId
        AND   Timestamp = insmsg.Timestamp
        AND   CopyId = insmsg.CopyId
        AND   keyCols = insmsg.keyCols;
if (sqlcode != 'no row found')
    // found a tombstone for the recieved row
    UPDATE insmsg.tableId
        SET CopyDelete = 'Y'           -- prevent capture
        Timestamp='0001-01-01.01'--prevent trigger
        WHERE keyCols = insmsg.keyCols;
    DELETE FROM insmsg.tableId
        WHERE keyCols = insmsg.keyCols;
```

FIG. 10 (continued)

11/17

```
while( UCTtime() < updmsg.new.Timestamp )
    sleep(updmsg.Timestamp - UCTtime()); //wait for future
checkDeltab = FALSE; implDel = 'N';
// Try for no conflict case
UPDATE updmsg.tableId
    SET nonKeyCols = updmsg.nonKeyCols,
        Timestamp = updmsg.new.Timestamp,
        CopyId = updmsg.new.CopyId,
        CopyDelete = 'N',
        ImplicitDelete = 'N'
    WHERE keyCols = updmsg.keyCols
    AND   Timestamp = updmsg.old.Timestamp
    AND   CopyId = updmsg.old.CopyId;
if (sqlcode == '1 row updated') // scenario 1: no conflict
    checkDeltomb = TRUE;
else // some kind of conflict, must look more closely
    //insert into deltomb if old CopyId not this copy
    if (updmsg.old.CopyId != thisCopy)
        INSERT INTO deltomb VALUES
            (updmsg.tableId, Updmsg.keyCols,
             updmsg.old.Timestamp, updmsg.old.CopyId);
    DECLARE updcur CURSOR FOR
        SELECT Timestamp, CopyId
        FROM updmsg.tableId
        WHERE keyCols = updmsg.KeyCols
        FOR UPDATE OF nonKeyCols, Timestamp, CopyId,
CopyDelete, ImplicitDelete;
    OPEN CURSOR updcur;
    FETCH updcur INTO Timestamp, CopyId;
    if (sqlcode == 'no rows found')
        // scenario 4: no matching key
        checkDeltomb = TRUE;
        INSERT INTO updmsg.tableId VALUES
            (updmsg.keyCols, updmsg.nonKeyCols,
             updmsg.new.Timestamp, updmsg.new.CopyId);
    else if (Timestamp < updmsg.new.Timestamp) ||
        (Timestamp == updmsg.new.Timestamp &&
         CopyId < updmsg.new.CopyId))
```

FIG. 11

12/17

```
//scenario 2: new Timestamp > existing row
checkDeltomb = TRUE;
if (CopyId == thisCopy)
    implDelete = 'Y';
UPDATE updmsg.tableId
    SET nonKeyCols = updmsg.nonKeyCols,
        Timestamp = updmsg.new.Timestamp,
        CopyId = updmsg.new.CopyId,
        ImplicitDelete = implDelete,
        CopyDelete = 'N'
    WHERE CURRENT OF updcursor;
else
    //scenario 3: new Timestamp< existing row
    if (CopyId == thisCopy)
        logConflict ("UPDATE", updmsg);
if ( checkDeltomb )
    SELECT 1 FROM deltomb
        WHERE tableId = updmsg.tableId
        AND    Timestamp = updmsg.new.Timestamp
        AND    CopyId = updmsg.new.CopyId
        AND    keyCols = updmsg.keyCols;
if ( sqlcode != 'no rows found' )
    // found a tombstone for the new row
    UPDATE updmsg.tableId
        SET CopyDelete = 'Y',          -- prevent capture
            Timestamp = '0001-01-01.01'--prevent trigger
        WHERE keyCols = updmsg.keyCols;
    DELETE FROM updmsg.tableId
        WHERE keyCols = updmsg.keyCols;
```

FIG. 11 (continued)

13/17

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (1201)		
//user update *upd X[1a] -> X[2a] ---> m2 (1202)	//propagated insert from 'a' m1 ---> ins X[1a] (1203)	
	//propagated update from 'a' m2 ---> X[1a] -> X[2a] (1204)	
	//user delete *del X[2a] ---> m3 (1205)	
//delete from 'b' m3 ---> del X[2a] (1206)		//delete conflict: not found m3 ---> deltomb add X[2a] (1207)
		//late arriving insert from 'a' //no match in deltomb m1 ---> ins X[1a] (1208)
		//update from 'a': no conflict m2 ---> X[1a] -> X[2a] (1209) //X[2a] found in deltomb del X[2a] (1210)

FIG. 12

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (1301)		//user insert *ins X[2c] ---> m2 (1302)
	//insert from 'a' m1 ---> ins X[1a] (1303)	
	//user update *upd X[1a] -> X[3b] ---> m3 (1304)	
//update from 'b' m3 ---> upd X[1a] -> X[3b] (1305)		//update from 'b': 3b>2c m3--> upd X[2c] -> X[3b] (1306) deltomb add X[1a] (1307) implDel X[2c] -> m4 (1308)
		//late insert from 'a' m1 ---> X[1a] in deltomb (1309)
	//user delete *del X[3b] -> m5 (1310)	
//delete from 'b' m5 -> del X[3b] (1311)		//delete from 'b' m5 -> del X[3b] (1312)
//late insert from 'c' m2 -> ins X[2c] (1313)	//late insert from 'c' m2 -> ins X[2c] (1314)	
//implDel from 'c' m4 ---> del X[2c] (1315)	//implDel from 'c' m4 ---> del X[2c] (1316)	

FIG. 13

15/17

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (1401)	//user insert *ins X[2b] ---> m2 (1402)	
		//insert from 'b' m2 ---> ins X[2b] (1403)
		//user update *upd X[2b] -> X[3c] ---> m3 (1404)
//update from 'c': 3b>1a m3--> upd X[1a] -> X[3c] (1405) implDel X[1a] ---> m4 (1406) deltomb add X[2b] (1407)	//update from 'c' m3 ---> upd X[2b] -> X[3c] (1408)	//user delete *del X[3c] ---> m5 (1409)
//delete from 'c' m5 ---> del X[3c] (1410)	//delete from 'c' m5 --> del X[3c] (1411)	
//late insert from 'b' m2 ---> X[2b] in deltomb (1412)	//late insert from 'a' m1 ---> ins X[1a] (1413)	//late insert from 'a' m1 ---> ins X[1a] (1414)
	//implicit delete from 'a' m4 ---> del X[1a] (1415)	//implicit delete from 'a' m4 ---> del X[1a] (1416)

FIG. 14

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (1501)		//user insert *ins X[3c] ---> m2 (1502)
	//insert from 'a' m1 ---> ins X[1a] (1503)	
	//user update *upd X[1a] -> X[2b] ---> m3 (1504)	
//update from 'b' m3--> upd X[1a] ---> X[2b] (1505)	//insert from 'c' m2 ---> upd X[2b] -> X[3c] (1506) implDel X[2b] ---> m4 (1507)	//update from 'b': $2b < 3c$ m3 ---> deltomb add X[1a] (1508)
//insert from 'c' m2 ---> upd X[2b] -> X[3c] (1509)		//implDel from 'b' m4 --> deltomb add X[2b] (1510)
//implDel from 'b' m4 ---> deltomb add X[2b] (1511)		//user delete *del X[3c] ---> m5 (1512)
//delete from 'c' m5 ---> del X[3c] (1513)	//delete from 'c' m5 ---> del X[3c] (1514)	//late insert from 'a' m1 ---> X[1a] in deltomb (1515)

FIG. 15

Copy A	Copy B	Copy C
//user insert *ins X[1a] ---> m1 (1601)		
	//insert from 'a' m1 ---> ins X[1a] (1602)	
	//user update *upd X[1a] -> X[2b] ---> m2 (1603)	
//update from 'b' m2--> upd X[1a] ---> X[2b] (1604)		//update from 'b': not found m2 ---> ins X[2b] (1605) deltomb add X[1a] (1606)
		//user delete *del X[2b] ---> m3 (1607)
//delete from 'c' m3 ---> del X[2b] (1608)	//delete from 'c' m3 ---> del X[2b] (1609)	//late insert from 'a' m1 ---> X[1a] in deltomb (1610)

FIG. 16